МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №1

по дисциплине

«Средства и методы защиты информации»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ Капранов С. Н.\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ Сорокин Е. А. \_\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_\_\_ 18-В1 \_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

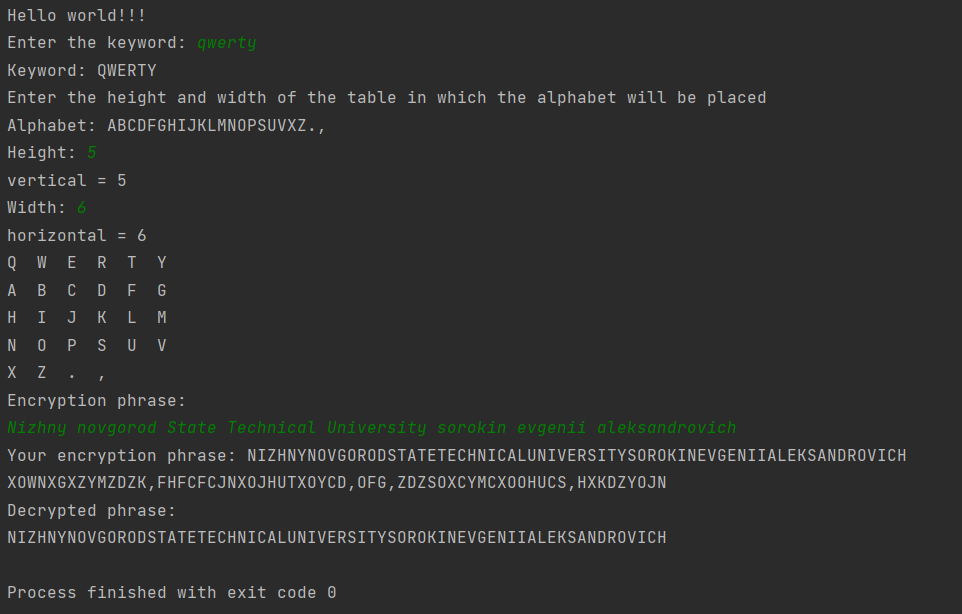
**Задание №1.3**

Реализовать алгоритм шифрования данных «Шифрующие таблицы Трисемуса».

**История и Алгоритм**

В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием "Полиграфия". В этой книге он впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово (или фраза). В таблицу сначала вписывалось по строкам ключевое слово, причем повторяющиеся буквы отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку. При шифровании находят в этой таблице очередную букву открытого текста и записывают в шифртекст букву, расположенную ниже неё в том же столбце. Если буква текста оказывается в нижней строке таблицы, тогда для шифртекста берут самая верхнюю букву из того же столбца**.**

**Пример работы программы**



**Код программы**

import java.util.Scanner;  
  
public class Main {  
  
 public static String noReplay(String title) {  
  
 StringBuilder bil = new StringBuilder(title);  
 int len = bil.length();  
 for (int i = 0; i< len; i++) {  
 char c = bil.charAt(i);  
 for (int j = len-1; j > i; j--) {  
 if (bil.charAt(j) == c) {  
 bil.deleteCharAt(j);  
 len--;  
 }  
 }  
 }  
 return bil.toString();  
 }  
  
 public static String newAlphabet(final String alphabet, final String title) {  
  
 StringBuilder bilAlph = new StringBuilder(alphabet);  
 int lenAlph = bilAlph.length();  
  
 StringBuilder bilTitle = new StringBuilder(title);  
 int lenTitle = bilTitle.length();  
  
 for (int i = 0; i < lenAlph; i++) {  
 for (int j = 0; j < lenTitle; j++) {  
 if (bilAlph.charAt(i) == bilTitle.charAt(j)) {  
 bilAlph.deleteCharAt(i);  
 lenAlph--;  
 }  
 }  
 }  
 return bilAlph.toString();  
 }  
  
 public static char[][] matrix(int row, int col, final String alphabet, final String title) {  
  
 char[][] matrix;  
 matrix = new char[row][col];  
  
 String noReplayTitle = noReplay(title);  
 String newAlphabet = newAlphabet(alphabet, title);  
  
 int lenNRT = noReplayTitle.length();  
 int lenAlpha = newAlphabet.length();  
  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 int position = (i \* col) + j;  
 if (position < lenNRT) {  
 matrix[i][j] = noReplayTitle.charAt(position);  
 } else if (position - lenNRT < lenAlpha){  
 matrix[i][j] = newAlphabet.charAt(position - lenNRT);  
 } else {  
 matrix[i][j] = ' ';  
 }  
 }  
  
 }  
 return matrix;  
 }  
  
 public static void printMatrix(char[][] matrix, int row, int col) {  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 System.out.print(matrix[i][j] + " ");  
 }  
 System.out.print("\n");  
 }  
 }  
  
 public static char search(char[][] matrix, int row, int col, char word) {  
  
 char res = ' ';  
  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 if (matrix[i][j] == word) {  
 if (((i + 1) == row) || matrix[i+1][j] == ' ') {  
 res = matrix[0][j];  
 } else {  
 res = matrix[i+1][j];  
 }  
 }  
 }  
 }  
 return res;  
 }  
  
 public static char unsearch(char[][] matrix, int row, int col, char word) {  
 char res = ' ';  
  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 if (matrix[i][j] == word) {  
 if ((i - 1) < 0) {  
 if (matrix[row-1][j] == ' ') {  
 res = matrix[row-2][j];  
 } else {  
 res = matrix[row-1][j];  
 }  
 } else {  
 res = matrix[i-1][j];  
 }  
 }  
 }  
 }  
 return res;  
 }  
  
 public static void replace(StringBuilder str, int index, char word) {  
 str.deleteCharAt(index);  
 str.insert(index, word);  
 }  
  
 public static String code(char[][] matrix, int row, int col, final String phrase) {  
  
 int lenPhrase = phrase.length();  
  
 StringBuilder bilPhrase = new StringBuilder(phrase);  
  
 for (int i = 0; i < lenPhrase; i++) {  
 replace(bilPhrase, i, search(matrix, row, col, bilPhrase.charAt(i)));  
 }  
 return bilPhrase.toString();  
 }  
  
 public static String uncode(char[][] matrix, int row, int col, final String codePhrase) {  
 int lenCodePhrase = codePhrase.length();  
  
 StringBuilder bilCodePhrase = new StringBuilder(codePhrase);  
  
 for (int i = 0; i < lenCodePhrase; i++) {  
 replace(bilCodePhrase, i, unsearch(matrix, row, col, bilCodePhrase.charAt(i)));  
 }  
 return bilCodePhrase.toString();  
 }  
  
 public static void main(String[] args) {  
  
 final String alphabetEN = "ABCDEFGHIJKLMNOPQRSTUVWXYZ.,";  
  
 System.out.println("Hello world!!!");  
  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("Enter the keyword: ");  
 String title = scanner.nextLine();  
 title = title.toUpperCase();  
 System.out.println("Keyword: " + title);  
  
 String noReplyTitle = noReplay(title);  
 String newAlphabet = newAlphabet(alphabetEN, title);  
  
 System.out.println("" +  
 "Enter the height and width of the table in which the alphabet will be placed");  
 System.out.println("Alphabet: " + newAlphabet);  
 System.out.print("Height: ");  
 Integer vertical = scanner.nextInt();  
 System.out.println("vertical = " + vertical);  
  
 System.out.print("Width: ");  
 Integer horizontal = scanner.nextInt();  
 System.out.println("horizontal = " + horizontal);  
  
 char[][] matrix = matrix(vertical, horizontal, alphabetEN, title);  
 printMatrix(matrix, vertical, horizontal);  
  
 System.out.println("Encryption phrase: ");  
 String phrase = scanner.nextLine();  
 phrase = scanner.nextLine(); //solving the problem  
 phrase = phrase.toUpperCase();  
 phrase = phrase.replaceAll("\\s+","");  
  
 System.out.println("Your encryption phrase: " + phrase);  
  
 String result = code(matrix,vertical, horizontal, phrase);  
 System.out.println(result);  
  
 System.out.println("Decrypted phrase:" + "\n" + uncode(matrix, vertical, horizontal, result));  
  
 }  
  
}